ATARI® PROGRAM exchange

DEVELOPER'S DISKETTE

INSTRUCTIONS

USER-WRITTEN SOFTWARE FOR ATARI PERSONAL COMPUTER SYSTEMS

APX-20034

DEVELOPER'S DISKETTE

INSTRUCTIONS
9/1/81

INTRODUCTION

OVERVIEW

This diskette contains preliminary versions of a few APX utilities, together with some demonstration and miscellaneous routines. Many of the programs contain rough spots. The contents of this diskette are subject to change without notice and such changes might not be reflected in this user manual. These routines are offered without follow-up support. The user manual does not conform to APX standards. In other words, the diskette and user documentation are offered strictly on as "as is" basis.

REQUIRED ACCESSORIES

ATARI BASIC Language Cartridge

OPTIONAL ACCESSORIES

ATARI 825 80-Column Printer
One ATARI Joystick Controller
Assembler Editor Cartridge

ATARI UTILITIES DISKETTE

This utilities diskette is distributed to outside software developers to facilitate their efforts. This software is not in the public domain and is not to be distributed. The utilities in this package are:

DOS	SYS	039	DOS 2.0S
DUP	SYS	042	DOS 2.0S
BUILD24		010	builds self-booting BASIC programs
DIV	SRC	026	an integer divide utility
BMUL	SRC	020	a signed integer multiply routine
SMUL	SRC	027	unsigned integer multiply routine
CHRGEN		049	a simple character set editor
~~SOUND~~		~~095~~	~~a sound editor~~
XREF		052	cross-reference program for BASIC
MASHER		045	compresses BASIC programs
RENUM	BAS	030	renumbers BASIC programs
FORMTR		004	formats LIST files from ASSEMBLER
M8TXT		005	BASIC mode 8 character print routine
BCDSAV		004	makes fixed-length number records
HOBBY1		008	simple player-missile graphics demo
HOBBY2		005	simple display list interrupt demo
SCRL19	ASM	104	full fine scrolling module
SCRL19	OBJ	011	object code
FIX	OBJ	042	a disk sector utility
SCRLH	DEM	004	simple horizontal scroll demo
MDIR	OBJ	002	gives disk directories from BASIC
SUPER	OBJ	007	a very fast sort routine
HORSE20	BAS	047	a character set animation demo
AUTORUN	SYS	001	the R: device handler
MDLI02	BAS	005	multiple display list interrupt demo
DEMO3	BOK	005	player priority trick
DEMO4	BOK	004	player as special character demo
DLISTA	DEM	005	alternating display list demo
SCRLF	DEM	003	simple fine scroll demo
SCRLV	DEM	004	simple vertical scroll demo

None of these programs are finished products; they all contain rough spots. If you make improvements please send a copy to our library.

CHARACTER EDITOR

This program is a character editor that makes it easier to take advantage of the redefinable character set capability of the Atari. It gives you the capability to edit, load, or save character sets.

The first menu option is to create or edit a character set. If you enter this option without first loading a character set, it will default to a blank character set. You edit characters with the joystick, selecting a pixel position with the stick and the status of that pixel (on or off) with the button. When you are done editing a character, you can allocate it to a character position with a keystroke indication. Fear not, most operations are prompted, so that you can pick your way through the program rather well. The only blooper is that the prompt for an I/O operation to the disk requires that you give the D1: prefix.

This program is usable but not at all as practical as IRIDIS's FONTEDIT program. I strongly urge you to buy and study the IRIDIS program if you want to do any character set work. If only it had a display list interrupt....

~~SOUND EDITOR~~

~~The sound editor helps you develop new sounds. It is not appropriate for developing tunes or jingles, or any long sound. It is designed for developing short sounds (1 second long) such as clangs, croaks, rattles, and other such nonsense. It only edits two of the four sound channels.~~

~~The program needs very little external documentation, as its title page describes the commands. The joystick response is slow, but you can use the 'fast' command to speed it up and then use the normal speed to fine tune your sound. You should read the hardware manual to get an idea of what the sound registers do.~~

~~Don't overlook the possibilities this program opens up. I have heard some very convincing sound effects created with it; it just takes a little imagination.~~

CROSS REFERENCE UTILITY

This utility provides a cross-reference of variables and constants in a BASIC program. It requires at least 40K of RAM, a printer, and a disk. The BASIC program to be cross-referenced should be on a diskette in SAVE format. The program first gives a count of the total number of variables used. For each variable, it lists all line numbers containing references to the variable. It also gives a count of how many times each constant is used. If an error occurs during printer output, you may recover by typing GOTO 3050.

RENUMBER

This program will renumber your BASIC program on disk. The target program must be on the diskette in LIST format. The program prompts you for the values it needs. The 'input device' will normally be D:progname. The 'output device' will normally be D:newname. The 'starting number' is the new starting line number. 'From' and 'to' are the beginning and ending line numbers of the section of code you want renumbered.

FORMATTER

This program will format the output of your Assembler/Editor cartridge so that you can get a list file that looks good out of your Atari 825 printer. Make the first instruction of your assembly code a .OPT NOEJECT. Then assemble the list file to the diskette. Then drop into BASIC and run this program. Respond to the name-prompt with D:progname. This program is designed for use with the Atari 825 printer, so good luck with anything else.

MAPSCROLL

Mapscroll is a demonstration program module that shows one way to use fine scrolling effectively. It creates a large map in BASIC's graphics 2 mode using a redefined character set. The map is 32x64 pixels in size, but only 10x20 pixels are displayed on the screen at any one time. The user can scroll the screen window across the map with the joystick. The program was written for easy integration into other packages.

Scrolling is achieved by coupling fine scrolling through the hardware fine scrolling registers with character scrolling by modifying the LMS bytes in the display list. The fine scrolling is straightforward; the character scrolling is somewhat more intricate. Each display byte in the display list has its LMS bit set. The following two bytes give the address of the display data. When the fine scrolling register overflows in the scrolling routine, the routine adjusts the bytes in the LMS addresses to point to the next character bytes. This adjustment is kept track of through a variable referred to as the offset.

The other trick in the program is the redefinition of the character set into a graphics character set. The technique is very powerful; very few of the available characters in the character set are used and yet the resulting map is very believable. The map could be made even more realistic by using the other characters. By changing the character set at appropriate times the program could produce a variety of effects.

The amount of system resource used is low. The module as written occupies 4K of RAM. This includes the map, the display list, the initialization routines, and the interrupt service routine that reads the joystick. Outside of this the program uses 4 bytes of page zero (two of which are available after initialization is complete) and seven bytes of page six. The interrupt service routine is very fast so it will not significantly slow whatever main program you plug it into. Space has been left inside the 4K block for additions and modifications. The program is not fully relocatable as there are four patches that must be made to relocate it; however, these patches are well documented and easy to do.

The easiest way to see this program in action is to BINARY LOAD it from DOS. The file name is SCRL19.OBJ. Then call it from BASIC with A=USR(27648). Plug a joystick into port 1 and scroll. At present the program does not interface well with BASIC; I have found that BASIC's cursor positioning gets lost. This reflects more on my laziness than on the program's tractability.

D. Yocum
4/9/80

CARE AND FEEDING OF "MASHER"

MASHER is a utility program writen in BASIC which compreses other BASIC programs. Since it is intended for internal use only, there are a few "features" which the user should be aware of.

1) All files are in LIST format.

2) Do not use lines 0-9 in the source program. MASHER will use these lines for it s own variable definitions.

3) Do not use the variables Q0 - Q999. MASHER uses these for constants.

4) Do not branch to REMark statements. Besides being bad pratice, MASHER has problems sometimes with this.

5) There is presently a bug with DATA statements. They are packed like any other statement. You will need to unpack these after MASHing. This bug will be fixed (someday).

WHAT IT DOES

MASHER will perform the following conversions on the source program

1) Removes all REMarks

2) Converts frequently used constants to variables.

3) Packs small lines together to form long lines.

NOTES

This program will usually save between 5% - 40%. Maximum compression occures when short lines are used in the source program. You must know the number of variables used in the source program. This can be obtained by running ~~SYMBOL~~ XREF.

PMMOVE

This program uses a machine language routine to move Players and Missiles. The routine is called by:

```
            USR(ADR(MOVE$),PMNUMB,XPOS,YPOS)
```

PMNUMB refers to the Player (0-3) or Missile (4-7) to move. The Missile number is determined by PMNUMB-4. XPOS and YPOS are the X and Y coordinates to place PM. A Relocator is used to make the PM Move routine wherever it may be placed in RAM by BASIC.

BINARY ROUTINE

This routine is included as part of the PMMOVE routine. BINARY loads or saves a binary file from BASIC.

```
     On entry  CMD=7    means load a file
               CMD=11   means save a file
               STADR=   the address to load or save a file from
               BYTES=   the number of bytes to save or load
               IOCB=    the IOCB to use
               FILE$=   file name to load

     On exit   ERROR=1  means successful load
               ERROR<>1 means it's an error status
```

BMUL

This routine implements BOOTHS ALGORITHM for multiplication of SIGNED binary numbers in TWO's-COMPLEMENT notation.

The MULTIPLIER is shifted to the right with the PRODUCT (as usual). Each CHANGE of the MULTIPLIER bits from zero to one causes the MULTIPLICAND to be subtracted from the PRODUCT. Each change from one to zero causes it to be added.

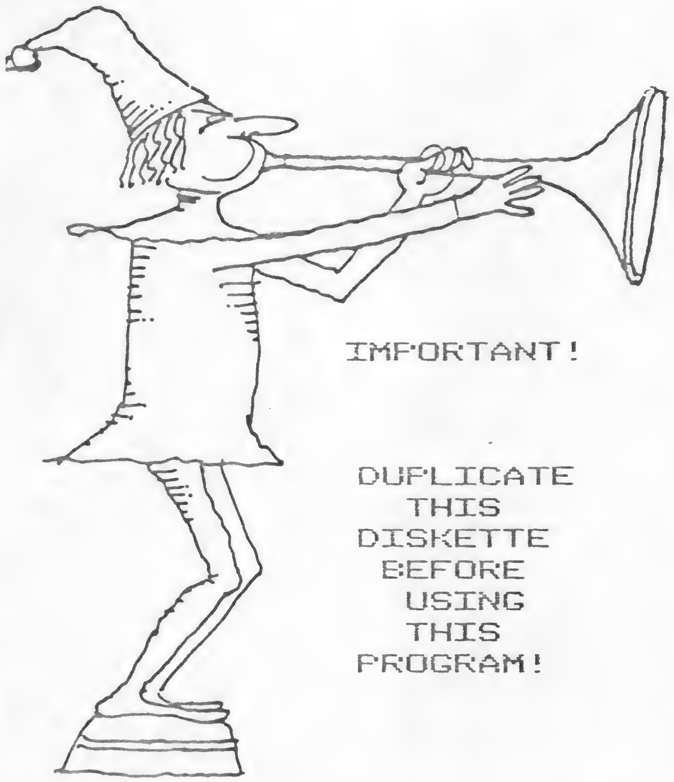
Like most signed arithmetic, it cannot be chained and is prone to overflow problems when given -32768, but it is smaller than the absolute-kludge wrapped around an unsigned mutiply, and not much slower (633-945 cycles).

```
     Enter with     A,Y = MULTIPLIER (A=MSB)
                    ACC = MULTIPLICAND

                     16 * 16  SIGNED MULTIPLY

     Exit with   ACC,MQ = 32 BIT PRODUCT
                    ACC = MSW
                     MQ = LSW

         ACC = 212
          MQ = $0CB
         ENT = $0CD
          SC = $0CF
           * = $600
```



IMPORTANT!

DUPLICATE
THIS
DISKETTE
BEFORE
USING
THIS
PROGRAM!

This APX diskette is unnotched to protect the software against accidental erasure. However, this protection also prevents a program from storing information on the diskette. The program you've purchased involves storing information. Therefore, before you can use the program, you must duplicate the contents of the diskette onto a notched diskette that doesn't have a write-protect tab covering the notch.

To duplicate the diskette, call the Disk Operating System (DOS) menu and select option J, Duplicate Disk. You can use this option with a single disk drive by manually swapping source (the APX diskette) and destination (a notched diskette) until the duplication process is complete. You can also use this option with multiple disk drive systems by inserting source and destination diskettes in two separate drives and letting the duplication process proceed automatically. (Note. This option copies sector by sector. Therefore, when the duplication is complete, any files previously stored on the destination diskette will have been destroyed.)

DIV

This routine is composed of two assembly subroutines: Unsigned Divide of 32/16 bits and Signed Divide of 16/16 bits.

```
     Unsigned Divide:

          Enter with A,Y = Divisor (A=MSB)
                  ACC,MQ = Dividend

          Exit with  ACC = Remainder
                      MQ = Quotient

     Signed Divide:

          Enter with A,Y = Divisor (A=MSB)
                     ACC = Dividend

          Exit with  ACC = Remainder
                      MQ = Quotient
```

SMUL

This routine is composed of a 16 * 16 Unsigned Multiply and a 16 * 16 Signed Multiply.

```
     Unsigned Multiply:

          Enter with A,Y = MULTIPLIER
                     ENT = MULTIPLICAND
                     ACC = ADDEND
                (ACC,MQ) = ([A,Y]*ENT)+ACC
                (This way for chained operaton)
                 556-748 Cycles

     Signed Multiply:

          Enter with A,Y = MULTIPLIER (A=MSB)
                     ACC = MULTIPLICAND

          Exit with  ACC,MQ = PRODUCT
                         ACC = MSW

          584-821 Cycles
```

M8TXT

M8TXT demonstrates how to mix text with graphics in BASIC mode 8. It plots the characters bit by bit onto the mode 8 screen. The technique can be adapted to any program using BASIC mode 8 displays.

BCDSAV

This program provides an alternative to the variable length records obtained when a program PRINTs values to the disk. Using the variable table values, it stores BCD values of numbers to the disk in fixed length records.

HOBBY1

This is a simple player-missile graphics demo. It sets up a player and then moves it around with the joystick. Since this is a pure BASIC program, the vertical motion of the player is too slow. An assembly language routine is necessary to get proper high-speed motion.

HOBBY2

This is a simple display list interrupt demo program. The bottom half of the screen changes from blue to pink.

BUILD24

This program creates an AUTORUN.SYS file that will start up your BASIC programs. It asks you for a BASIC command; the command you enter will be in the AUTORUN.SYS file and will be executed on powerup. Normally your command will be of the form RUN"D:PROG.BAS". It must be less than 128 characters long. Remember to put in the terminating double quotation mark (").

MDIR

This program is an object file that can be called from BASIC to put the disk directory onto the screen. To use it, you must first load it into RAM with the BINARY LOAD command (L) in the DOS. Call it from BASIC with A=USR(1536).

AUTORUN.SYS

This program is an RS-232 handler file. It allows you to use the R: device. It is booted in automatically on powerup.

HORSE20.BAS

This is the running horse demo. It demonstrates the power of character set graphics and animation.

SCRLH.DEM

This is a simple horizontal coarse scroll demo.

SCRLV.DEM

This is a simple vertical coarse scroll demo.

SCRLF.DEM

This is a simple fine scroll demo.

MDLI02.BAS

This is a multiple display list interrupt demo. It puts gobs of color onto the screen. Do not be alarmed if the screen goes black; it takes several minutes to finish. Once it is running observe how keypresses affect the display. Also note the greatly reduced computation speed of BASIC. With so many interrupts happening, the 6502 has little time for other activities.

DEMO3.BOK

This program demonstrates a technique for increasing the resolution of a stationary player by hiding it behind a playfield cutout.

DEMO4.BAS

This program shows how a player can be used as an extended character.

DLISTA.DEM

This program demonstrates the alternating display list technique.

ATARI PROGRAM EXCHANGE

REVIEW FORM

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many software authors are willing and eager to improve their programs if they know what users want. And, of course, we want to know about any bugs that slipped by us, so that the software author can fix them. We also want to know whether our documentation is meeting your needs. You are our best source for suggesting improvements! Please help us by taking a moment to fill in this review sheet. Fold the sheet in thirds and seal it so that the address on the bottom of the back becomes the envelope front. Thank you for helping us!

1. Name and APX number of program ______________________________

2. If you have problems using the program, please describe them here.

____

____

____

3. What do you especially like about this program?

____

____

____

4. What do you think the program's weaknesses are?

____

____

____

5. How can the catalog description be more accurate and/or comprehensive?

____

____

6. On a scale of 1 to 10, 1 being "poor" and 10 being "excellent", please rate the following aspects of this program?

_____ Easy to use
_____ User-oriented (e.g., menus, prompts, clear language)
_____ Enjoyable
_____ Self-instructuve
_____ Useful (non-game software)
_____ Imaginative graphics and sound

7. Describe any technical errors you found in the user instructions (please give page numbers).

8. What did you especially like about the user instructions?

9. What revisions or additions would improve these instructions?

10. On a scale of 1 to 10, 1 representing "poor" and 10 representing "excellent", how would you rate the user instructions and why?

11. Other comments about the software or user instructions:

```
 -----
|     |
|STAMP|
|     |
 -----
```

ATARI Program Exchange
P.O. Box 427
155 Moffett Park Drive, B-1
Sunnyvale, CA 94086

[seal here]